Engineering with Computers (1996) 12: 1–15

Engineering with Computers

Variant Design for Mechanical Artifacts: A State-of-the-Art Survey

J. E. Fowler

National Institute of Standards and Technology, Factory Automation Systems Division, Building 220, Room A127, Gaithersburg, Maryland 20899, USA

Abstract. *Variant design refers to the technique of adapting existing design specifications to satisfy new design goals and constraints. Specific support of variant design techniques in current computer aided design systems would help to realize a rapid response manufacturing environment. A survey of approaches supporting variant design is presented. Capabilities used in current commercial computer aided design systems are discussed along with approaches used in recent research efforts. Information standards applicable to variant design are also identified. Barriers to variant design in current systems are identified and ideas are presented for augmentation of current systems to support variant design.*

Keywords. Analogical reasoning; Case-based design; Computer aided design; Design knowledge; STEP

1. Introduction

Variant design is a technique supporting retrieval of an existing design specification for the purpose of adapting that design specification for use in the design of a new but similar artifact. Design retrieval mechanisms may range in complexity from manual search to automatic identification of similar designs based on specifications such as desired functionality. Once an existing design specification is identified, a number of techniques may be employed to adapt the design. Design adaptation techniques can range in sophistication from manual modification to automatic modification based on specifications of design goals and constraints.

The sophistication of mechanisms for design adaptation is related to the sophistication of design retrieval techniques; they are linked through abstracted representations of artifact functionality, behavior, and so on. One can expect that a system capable of automatically adapting a (known) design specification successfully will utilize underlying representations which also enable sophisticated design retrieval strategies. Conversely, a system that provides rudimentary design retrieval based on manual review is unlikely to provide design adaptation capabilities beyond manual parametric modification. Near-term capabilities in design systems lie somewhere in between these two extremes.

Members of the National Center for Manufacturing Science's (NCMS) Rapid Response Manufacturing (RRM) consortium have identified variant design as a technology warranting further development for implementation in their companies. This state-of-the-art survey is intended to provide an understanding of variant design and therefore stimulate further work in the context of rapid response manufacturing. Specifically, this assessment focuses on mechanical artifact design representation/retrieval techniques available today in commercial computer aided design (CAD) systems and on relevant techniques under investigation in recent research. Information standards under development that are pertinent to these areas are also discussed. The final section of this paper presents some ideas on incorporating research results into current CAD systems.

Correspondence and offprint requests to: J. E. Fowler, National Institute of Standards and Technology, Factory Automation Systems Division, Building 220, Room A127, Gaithersburg, MD 20899, USA.

2. Current CAD Systems: Capabilities and Usage

This section investigates specific aspects of current commercial Computer Aided Design systems that are relevant to variant design. The geometric modeling capabilities (i.e. the types of curves and surfaces supported) of commercial CAD systems are not considered to be relevant to this investigation. It can be reliably assumed that the geometric and solid modeling capabilities of commercial CAD systems are sufficient to provide an unambiguous representation for the shape of an artifact.

There is a number of capabilities offered by commercial CAD system vendors which may be considered as functions enabling variant design; however, these capabilities do not by themselves provide a variant design environment. 'Feature-based modeling' and 'parametric design' are techniques that can assist a designer in developing new artifacts that are related to but distinct from those previously designed. Each CAD vendor naturally provides somewhat differing capabilities, or may refer to apparently equivalent capabilities by different names from those of another vendor. Yet there is sufficient commonality between systems to discuss these capabilities generically.

2.1. Feature-based Modeling

This capability allows the user to create designs by combining instances of three-dimensional shapes representing commonly recognized forms, e.g. holes, slots, ribs, bosses, keyways, and so on. The ability to directly instantiate such features with varying attribute values is typically a built-in function of a system; common features of form are considered as a vendor-provided library of features. Most systems allow the user to augment the built-in library with definitions for their own collection of features (i.e. 'user-defined features') which can be manipulated and used in the same way as the vendor-provided library of features.

On the one hand, a feature-based modeling approach can be considered as a shorthand (or 'macro' operation) approach to defining aspects of the design shape. This is particularly true if after instantiation a feature loses some aspects of its original characteristics following neighboring changes in the design shape. Consider a 'blind-hole' feature with the characteristics that it creates a void to a specified depth from some reference and that the void is bounded by a cylindrical surface and a planar surface (Fig. 1). At some point after instantiation of a 'blind-hole', the designer modifies the location of the reference that determines the depth of the 'blind-hole'; this modification has the side-effect of changing the 'blind-hole' to a hole that penetrates the artifact (see Fig. 2). Will the CAD system prevent the designer from making the change that modifies the original characteristics of the 'blind-hole'? If not, will the designer be notified that the 'blind-hole' no longer exhibits its original properties? Will the CAD system automatically identify the modified features as a 'thru-hole'? If a particular CAD system does none of these, then perhaps it is accurate to describe that system's feature-based modeling capability as simple macros for series of operations that are meaningful in certain contexts. Conversely, a CAD system that detects the side-effect and acts on it in a way that is useful to the designer is providing more than a built-in library of familiar macro operations: it is maintaining the designer's prescribed relationships between aspects of the design. Examples of feature-based modeling in current commercial CAD systems can be found at both ends of this spectrum.

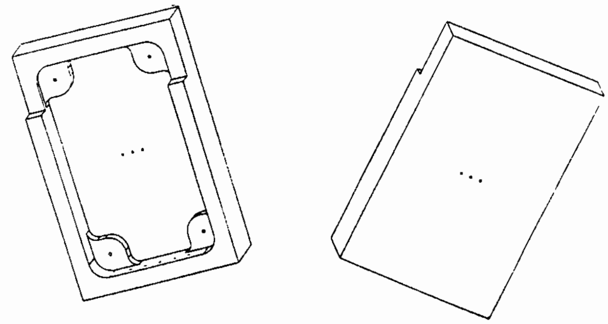

Fig. 1. 'Blind-hole' features. Looking at the view of the housing appearing at left, a hole is apparent in each of the four mounting pads located in the central pocket of the housing. Each of the four holes is designated as a 'blind-hole'. A specific numeric value controls the holes' depth with respect to the top surface of the mounting pads; the holes are not intended to penetrate the bottom surface of the housing. The view appearing at right shows that the 'blind-holes' do not penetrate the housing.

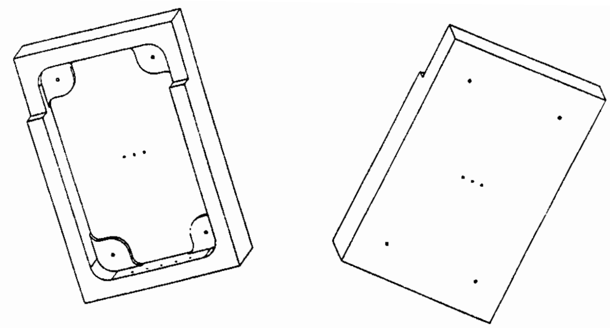

Fig. 2. Indirectly modified 'blind-holes'. Comparing the view appearing at left here with that in Fig. 1, it should be apparent that the thickness of the mounting pads has been decreased. Since the depth of the hole in each mounting pad is referenced from the top surface of the pad, the holes extend below the housing. The view on the right shows the four holes penetrating the bottom of the housing. The 'blind-hole' features are now effectively 'thru-holes', although the CAD system maintains them by their original designation.

2.2. Parametric Design

This capability allows the user to instantiate designs by supplying parameter values to implicitly or

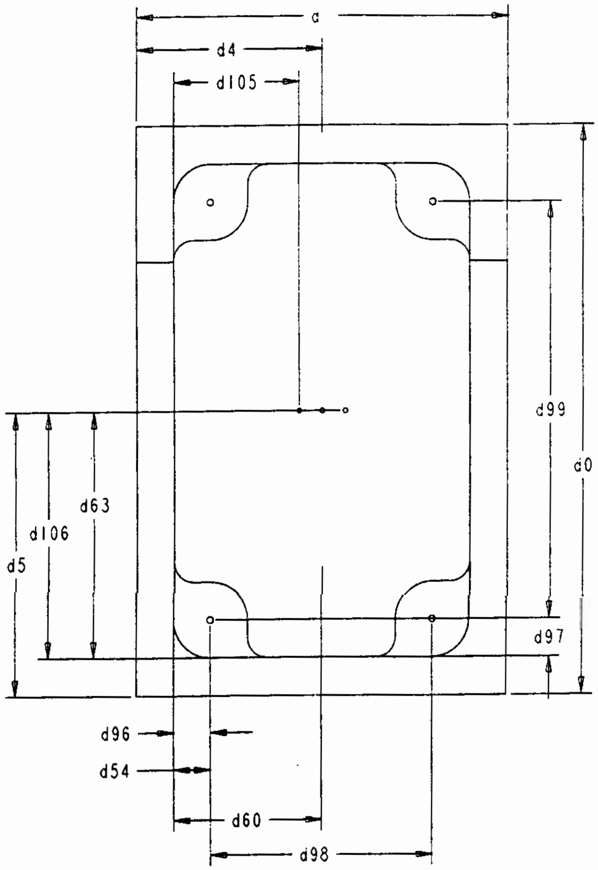

Fig. 3. Symbolic dimensions. This orthographic view of the housing part is augmented with annotations representing some of the dimensions established in the development of the part design. Once symbolic dimensions are identified, the designer can replace implied values with specific values or equations.

RELATION	PARAMETER	NEW VALUE
d4 = d1/2	d4	4.00e+00
d5 = d0/2	d5	6.00e+00
d0 = d1*1.5	d0	1.20e+01
d97 = d96	d97	8.00e-01
d96 = d54	d96	8.00e-01
d98 = 2*(d60-d54)	d98	4.80e+00
d99 = 2*(d63-d97)	d99	8.80e+00
d106 = d63	d106	5.20e+00
d105 = d60-0.5	d105	2.70e+00

Fig. 4. Parametric relations. This table is a reproduction of one maintained by the CAD system that was used to design the housing part. The left column in the table shows the system of equations that has been established between particular dimensions of the housing part by the designer. The middle column indicates which parameter is the dependent variable in the relationship. The right column shows the current value for the dependent variable obtained by evaluating the system of equations in conjunction with explicit and implicit dimension values.

explicitly identified attributes of the design (see Fig. 3); it typically works in conjunction with the capability of feature-based modeling. A trivial example of such a capability would be the provision of a diameter parameter for a hole. A typical example is parameterization of the positioning of a pattern of features with respect to another feature. These systems also allow the user to establish equations relating the values of parameters to others, e.g. the depth of a slot is half of its length, or that a pattern of holes all have the same diameter (see Fig. 4).

Aside from numeric assignment of values to attributes, systems may also maintain relations between geometric aspects of the artifact. These relations may also be thought of as design parameters, e.g. that the user has specified that two planar surfaces are parallel, or that two holes are coaxial, or that an edge has been replaced by a fillet and the resulting surface is tangential to the surfaces that were neighbors to the edge that was replaced. Systems that support this kind of parametric information maintain this information in the same way that numeric parameters are maintained, i.e. until the user provides new information or until maintenance of the relationship results in some geometric inconsistency.

The net effect of representation and maintenance of parametric information is that the artifact design is constrained by the established parametric relations. These parameters result in a system of equations that the CAD system must solve in order to evaluate the shape of the artifact. Whether or not the CAD system solves these equations sequentially in the order that the designer has supplied the parametric information plays a role in determining how much flexibility the designer has in specifying parameters. A system may be able to evaluate the parametric relations only in the order in which they were established by the designer; thus the designer must be careful to fully constrain the shape of the design (via the parametric information supplied) at each stage of its evolution without over-constraining it. On the other hand, a system may have the ability to evaluate the parametric relations in an order-independent fashion; this capability is often referred to as 'variational design'. The phrase 'variational design' does not connote variant design. Variational design is the marketing phrase used by the vendors of several parametric CAD systems to describe the capabilities of those systems allowing order-independent evaluation of parametric equations.

With variational design, the designer need not be concerned that the sequence of parametric relations establishes a fully constrained design. Instead, the CAD system allows the design to remain under-constrained as it evolves and may in fact make assumptions about the design in order to evaluate the design shape, e.g. lines that were drawn apparently parallel are assumed to be parallel and yield parallel planar surfaces when swept in a linear extrusion. Nevertheless, at some point in time the CAD system will require that the design shape be fully constrained. The CAD system will allow the designer to continue to embellish an under-constrained design until some new design parameter relation contradicts a relationship that the system had implicitly assumed – at that point the CAD system will force the designer to establish a consistent set of relationships. The principal advantage offered by variational design is flexibility in terms of how long relationships between aspects of the design shape can remain under-constrained while the designer continues to synthesize the design.

2.3. Relevant Standards

Participants in the International Organization for Standardization (ISO) have endeavored to codify many aspects of CAD data to facilitate information exchange between systems. The Standard for The Exchange of Product model data (STEP) includes specifications describing what data are to be exchanged, what the context of exchange is, and how the data are to be communicated [1]. Among the specifications STEP includes are those describing product shape through form features based on geometry and topology and other shape aspects [2, 3].

The STEP form features model 'characterizes and represents shapes that are of broad industrial interest' [3]. The features described are specifically intended not to have any industrial connotation, i.e. there is no context for how an application would use any features described using the model. For example, some semi-automated manufacturing planning applications associate particular material removal methods with particular feature types; interpretations of this nature are explicitly avoided by the STEP form features model. Designation of the context and usage of these feature descriptions are left as the job for other specifications in STEP known as Application Protocols.

It is important to note that in STEP, the Application Protocols are the specifications that CAD system vendors implement in order to provide conforming data exchange capabilities. Many current CAD systems supporting feature-based modeling use internal feature representations that are compatible with those described in the STEP form features model [4]. Here, compatibility does not necessarily mean that a vendor's internal representations are precisely the same as those described in STEP, but rather that those described by STEP are derivable from a vendor's product. At this time it is difficult to predict whether vendors' products will be compatible with Application Protocols that make use of form features, since such specifications are still progressing through the standards process. The STEP working groups are also currently considering a new work item covering representations for parametric relations, but this effort is very much in its infancy.

2.4. Variant Design and Current CAD Systems

The techniques of feature-based modeling and parametric design allow designers to easily modify certain characteristics of existing artifact designs to meet new specifications. It is clear how these techniques can be useful: to scale the proportions of a mounting bracket to support a larger load, to change the radial placement of bolt holes when a higher torque must be accommodated, to change the pitch of a worm gear in correspondence to a change in the number of teeth in a pinion gear, and so forth. These techniques allow a designer to instantiate families of designs with each design structurally related to its siblings but differentiated by parametric relationships satisfying particular design objectives.

Yet current CAD systems exhibit shortcomings for a variant design environment. The first shortcoming is design retrieval. The only mechanism that CAD systems provide to designers to retrieve existing designs is by part or assembly name. These names are textual, they may have been automatically generated by the CAD system itself or specified by the designer, and typically correspond to a file name convention supported by that host computer platform. While design rules in force at a particular company may provide naming guidelines to designers, there is no guarantee that a given textual name has any relationship to the functionality, intended use, or context for the design.

Finding an existing design produced using a current CAD system without any mechanism other than a part or file name becomes a chore not significantly different from locating a design drawing in a storage room of archived (shelved) drawings. Like its paper-based predecessor, a CAD system and the host computer operating systems provide features for organizing design files. Designs may be indexed by ownership to a designer, as member components in an assembly hierarchy, as the root of an assembly, as a particular design project, by revision status, and so forth, depending on the features offered by the particular CAD system. But such indices may offer only vague guideposts to a designer who is wondering whether any existing compressor housing designs provide adequate clearance for a redesigned compressor. Finding a selection of existing compressor housing designs may take longer than designing a satisfactory housing from scratch ('Where is that housing we used on the T2 project? Is it in Beth's directory? ... Is this the released part? I thought it had a flat mounting plate...').

After finding existing designs that may be applicable to a current problem, a designer will have to evaluate whether any of the existing designs satisfy the new design requirements. At the same time, the designer will consider whether any of the existing designs can be easily modified to meet the new requirements if none of the existing designs is immediately satisfactory. Such evaluation highlights another shortcoming of current CAD systems for variant design – representation of design rationale, intent, or justification. To a degree, aspects of these kinds of design information are represented. The fact that a feature is a 'thru-hole' is construed to mean that the designer intended the hole to penetrate the part. The parametric relation governing the placement and size of bolt holes in a radial pattern does indirectly indicate the torque being counteracted. These parametric and feature-based modeling capabilities are often advertised by vendors as means for 'capturing design intent'. The explicit design intent conveyed by these representations offers insight as to how a part was designed, but information about why it was designed the way it was, or what roles particular aspects of the design play in achieving the artifact's design goals, is still implicit. So the designer who was wondering why a compressor housing mounting plate was not flat may not be able to infer from the CAD system representation that the reason was to reduce vibration.

The more information that is represented in a design regarding rationale, intent and justification, the better equipped the designer will be to evaluate whether or not an existing design is suitable (or adaptable) for a new purpose. A designer can interpret the parametric equation captured in a CAD system relating applied torque to the configuration of a bolt hole pattern and recognize what options there are to change the design parameters to accommodate new requirements for a lower torque and reduced material weight. A computer program to do the same will require significantly more information about a design than features and parametric relations.

3. Variant Design: Related Research

A number of research efforts are relevant to the topic of variant design. Efforts in analogy-based problem-solving and analogical reasoning focus on how to map existing problem solutions to new problems. Case-based reasoning encompasses aspects of analogical problem-solving while more closely examining the issues of what aspects of existing problem solutions need to be represented in conjunction with how to select existing problem solutions relevant to the problem at hand. Both analogical problem-solving and case-based reasoning exhibit aspects of machine learning when these approaches seek to extend the problem-solving capabilities of a system by storing successfully adapted solutions for later use.

In addition to design, these research techniques have been applied to a variety of problem domains. Theorem-proving [5], mediation [6], customer service telephone support [7, 8], recipe planning [9], meal planning [10] and part layout for autoclave processing [11] are among the areas that have provided suitable problems for exploration. The following section will focus on the research efforts in the context of design; however, many of the fundamental research techniques have arisen from investigations in other problem domains and will be discussed as appropriate.

3.1. Analogical Reasoning Applied to Design

Solving new problems by analogy to existing solutions is a familiar technique used by humans in a wide range of problem domains. A formal definition of analogical problem solving due to Carbonell [12] is:

> 'Analogical problem solving (reasoning) consists of transferring knowledge from past episodes to new problems that share significant aspects with corresponding past experience – and using the transferred knowledge to construct solutions to the new problems.'

In the parlance of the literature, a new problem to be solved using analogical techniques is generally referred to as the *target* problem (or simply as the target). The past episodes from which analogies are formed are *bases* or *sources*. Hall [13] surveys computational approaches to analogical reasoning and provides a framework with which to compare approaches. Hall's framework describes four basic process components that characterize computational approaches to analogy:

1. *recognition* of a candidate analogous source, given a target description;
2. *elaboration* of an analogical mapping between source and target domains, possibly including a set of analogical inferences;
3. *evaluation* of the mapping and inferences in some context of use, including justification, repair, or extension of the mapping;
4. and *consolidation* of the outcome of the analogy so that its results can be usefully reinstated in other contexts.

Recognition strategies range in complexity from explicit identification of the base from which to form the analogy [14] to mechanisms for searching through sources based on similarity [15] or other criteria. In *elaboration*, the general problem is to restrict a (conceivably large) space of possible mappings between elements of the source and target to a smaller space of potentially useful mappings. The mapping space can be constrained by identifying known mappings which can be reused and/or incrementally extended. Techniques for determining which mappings are reusable or extensible include finding those that preserve the relational structure of the source description [16], finding those that preserve selected semantic categories [17] and finding those that preserve information relevant to the current reasoning context [18]. A mapping that can be reused satisfies the purpose of the analogy directly; a mapping that is extended must be evaluated for applicability to the target. In *evaluation*, the tentative inferences may be tested against expectations of the target, justified in the context of the target, and possibly repaired if found to be inappropriate. Future performance of the reasoner can be improved by *consolidating* the source, target and evaluated inferences for later use, i.e. 'learning by analogy'. Consolidation techniques can include recording the target and outcome [19, 20], forming inductive summaries over the source and target [12, 15] and recording failed analogies [6, 19].

Navinchandra *et al.* [21] outline requirements that a computational model for analogical reasoning must address to support the basic process components. The model should be knowledge-based so that correspondences between the target and sources can be recognized, and the model must support efficient organization, retrieval and consolidation of experiences. The model should enable retrieval of analogies based on matching to different levels of detail and thus there should be mechanisms to abstract analogies to different levels. The model should enable elaboration and evaluation of the analogical mapping based on the intended purpose of the source analogies and the causal network of relations in the sources. Finally, the model should enable effective use of computing resources.

Transformational analogy and derivational analogy are elaboration techniques introduced by Carbonell [12, 15]. Many later research efforts were influenced by Carbonell's work and exhibit characteristics of the two techniques. In transformational analogy, a space of transformation operators is searched and applied in order to extend a partial mapping. Thus the source analogy is transformed to solve the target problem. Transformational analogy is limited by the assumption that problems sharing similar characteristics also share problem-solving strategies.

The STRUPLE system [22, 23] used the transformational analogy approach in preliminary structural design of buildings. In this system, the aspects of the source transferred were elements from a set of design elements (e.g. beams, columns, braced frames) which constituted a design vocabulary for structural design. Given specifications for a building to be designed, STRUPLE matched existing building design solutions based on selected similarity criteria and selected elements of the design vocabulary for the new design. STRUPLE operated interactively, providing the user with the means to revise the similarity criteria used to match existing building designs, as well as to add to or delete from the set of design elements extracted from the matched designs. STRUPLE was not intended to perform synthesis of structural building designs, but rather to be a tool for retrieving relevant design experience and as a preprocessor for a more comprehensive design process.

Derivational analogy addresses the limitations of transformational analogy by transferring the reasoning process exhibited by the source to the target problem. For example, FIRST [24] was developed to redesign structural beams using a knowledge base of design plans for existing beams. The specifications for the beam to be designed were presented to FIRST in the form of constraints that the beam was to satisfy. Given the design constraints, FIRST would generate a system of equations describing the behavior of the desired beam; if the initial design specifications invalidated these equations, the knowledge base of existing design plans was searched to find beam design plans that were sufficiently similar to the desired beam. Similarity matching was based on the relationship between constraints that were violated/ satisfied in the desired beam as compared with the corresponding constraints contained in the existing design plans. The selected design plans provided the source of actions which could be considered and applied according to their relevance to the current problem. FIRST was limited in the sense that it used fairly simple techniques to find relevant design plans and actions from those plans. However, it did illustrate how design modifications from analogous designs could be automatically selected and combined to arrive at a satisfactory design solution.

Mostow [25] provides a description of the issues arising from the application of derivational analogy to design by examining four systems implemented using this approach. The four systems all replay existing design plans in the design of complex artifacts: POPART [26], REDESIGN [27], BOGART [28] and ARGO-V† [29]. POPART was used to transform software specifications into executable programs; the other three systems were all used in the domain of digital circuit design. Mostow discusses how each of the four systems addresses the following issues in replaying a design plan:

1. Representation: What information about the original design decisions is needed in order to replay them, and how should it be expressed?
2. Acquisition: How can this information be captured?
3. Retrieval: Given a problem, how can relevant previous designs be found?
4. Correspondence: Which objects, goals, constraints, etc. in the new design correspond to which ones in the old design?
5. Appropriateness: When should a given plan or plan step be replayed?
6. Adaptation: How can a previous plan be altered to fit a new problem?
7. Partial reuse: Which parts of a plan can be replayed by themselves?

Mostow characterized ARGO-V 'as the most complete system to date for design by derivational analogy'. It refines functional specifications describing a digital circuit's behavior into a description of the synthesized circuit's structure. ARGO-V's design knowledge base contains frame definitions, frame instantiations, assertions and rules. Frame definitions are based on VHDL [VHDL is the VHSIC hardware description language] entities describing interface bodies (which define an entity's externally visible ports and parameters) and one or more architectural bodies (which define entities in terms of behavior and structure). Frame instantiations refer to primitive library components such as transitors, logic gates and the like. Assertions describe library component slot values. Rules perform refinement steps, i.e. transformations (conversion of signal assignment statements into simpler or more convenient forms), decomposition (to group logically related signal assignment so that they can be treated as independent subproblems), or instantiation (of library components).

A design plan in ARGO-V is presented as a database of assertions stored as slots of frames. A truth maintenance system implicitly represents the relationship between instantiated rules forming a rule dependency graph. Rule dependency graphs are compiled into macro-rules so that a design plan can be replayed by executing the corresponding macro-rule. Design plans are stored at increasing levels of abstraction so that inexact analogies can be executed. As macro-rule abstractions are computed, the abstractions are partially ordered in terms of their abstractness; this ordering reduces time spent searching for relevant rules to execute. Macro-rules are retrieved if their preconditions match the specifications of the new problem; all consistent sets of bindings for the parameters used in the macro-rule are found with each binding, leading to a rule instantiation for consideration. ARGO-V retrieves and executes rules automatically but can also operate interactively with the user, specifying rule preferences and priorities for execution.

Use of ARGO-V shows that it could reduce the amount of time spent solving new design problems after learning from an original problem, that it could apply inexact analogies towards the solution of new design problems, and that the quality of design (as measured by the number of components required) improved after learning from analogous designs. On the other hand, success with ARGO-V is limited in that the search space of macro-rules grows as ARGO-V learns and can lead to increased solution time when the system's initial rule set can be applied directly [this increase can be evident despite reduction in the search space by elimination of macro-rules based on their abstractness]. Abstracted design plans are used without modification; this limits the domain of designs that can be solved/learned. Finally, the system's control strategy is biased towards problem-solving time reduction but there are other criteria for desirability which may be important as well.

† It is worthwhile to note that ARGO refers to the environment used for building knowledge-based problem-solving systems that improve with use; ARGO-V is the name of the application that was constructed in ARGO for Very Large Scale Integrated (VLSI) circuit design.

3.2. Case-based Reasoning Applied to Design

While analogical reasoning approaches primarily focus on how to apply base analogies to target problems, case-based reasoning approaches broaden the focus to address issues of how to select, represent and organize analogies, i.e. cases. Two systems are of particular interest here, CADET [30] and KRITIK [31, 32]. CADET solves engineering design problems using representations that capture the relationship between function, structure and behavior in a case-base. KRITIK solves engineering design problems using a case-base of designs represented by components and substances, their relationship and behavior.

CADET performs conceptual design by synthesizing a device from pieces of design problem solutions ('snippets') accessed from previous design cases. The initial case-base for CADET was populated with cases spanning a wide range of engineering domains (e.g. hydraulic, mechanical, electrical), with cases offering different structural realizations of the same device behavior, and with cases that described families of devices with differing performance characteristics. The input design specifications to CADET consist of functional and behavioral characteristics of the desired device along with physical constraints on the device. CADET produces a conceptual schematic describing a device which satisfies the input specifications.

CADET's case memory stores design cases in terms of function, behavior and structure along with the relationships between those aspects of each design. The case memory is organized to support indexing by linguistic descriptions of devices, functional block diagrams, device behavioral abstractions, qualitative states, as well as structural and performance features. Device behavioral abstractions are characterized as influence graphs which relate qualitative relations between variables of interest (e.g. orifice size and flow rate) [33]. The design specification input to CADET is transformed into an index graph; the desired device's index graph is then matched against the devices in case memory [34]. Since there may not be a one-to-one mapping between indices characterizing the desired device and those in the case memory, index transformations may be performed on the desired device's index graph. These index transformations preserve the specified behavior of the device while improving the likelihood of finding a suitable match in the case memory. Such behavior-preserving transformations are based on knowledge (known or hypothesized) about the physical laws and principles that are going to govern the design solution.

Case matching in CADET can result in multiple alternatives to consider. Alternatives stem from influence subgraphs which can satisfy sub-behaviors of the desired device. Choosing between alternatives is controlled by criteria such as total cost, weight and a heuristic evaluating the complexity of device synthesis given the set of components under consideration. Case adaptation can be performed for material selection, i.e. re-evaluating the reasons why a material was used in a precedent case and determining whether the same factors apply in the current problem.

Experience with CADET showed how design sub-cases could be combined to create new devices and how representations for device behavior could support case matching, retrieval, combination and adaptation across a variety of engineering domains. Central to CADET's capabilities is the approach of describing device behavior through influence graphs. CADET's behavioral model is independent of the device structure used to achieve the behavior; this is in contrast to KRITIK which correlates device structure with the behavior achieved.

KRITIK is given a description of the functions required of a device with the goal of producing an output specifying a design for a device that can deliver the desired functions. Knowledge of previous design experiences is organized in a case memory. A design case in KRITIK comprises the design's structure, the functions that the design can deliver and a pointer to a structure–behavior–substance model. The structure–behavior–substance model represents how the design's structure achieves its functions and provides part of the knowledge necessary for KRITIK to perform design adaptation.

Design cases in KRITIK are organized according to the functions that they deliver; if a design delivers more than one function it is multiply undexed by each of the functions. KRITIK uses the input specifications describing desired functionality as the means to find designs in the case memory that are most similar; cases delivering functionality closest to that desired are potentially the easiest to adapt. Underlying KRITIK's case function representation capabilities is a component–substance model capturing the structure and functioning of physical devices.

In the component–substance model, the structure of a device is represented as components (e.g. a pipe), substances (e.g. water) and the structural relations between the components and substances (e.g. containment). Although the component–substance model can support various types of functions, KRITIK focuses on state transformation functions. Such functions transform an input behavioral state to an output behavioral state, e.g. cooling a substance from one temperature to another. This component–substance model is used as the vocabulary in a behavioral representation language describing device functions. Function schemas in the behavioral representation language describe the input and output behavioral states of a device, the internal causal behavior responsible for the state transformation, along with the internal and external conditions enabling the function [35].

The function schema representation used in KRITIK allows for case retrieval based on functional indices and permits the identification of similar design cases based on partial matches. Domain-specific heuristics are used to resolve situations where the cases retrieved are deemed to be equally similar (i.e. they differ from the target functionality by an equivalent number of functional indices). Adaptation of a retrieved case occurs based on the functional differences between the retrieved case and the target specification. KRITIK maintains a family of modification plans appropriate to resolution of specific functional differences; these are applied to perform case adaptation. As with selection of the most relevant and easily adaptable case, domain-specific heuristics are used to determine the order in which modification plans should be applied when multiple functional differences must be resolved.

KRITIK's design domain is limited to devices whose functions can be characterized in terms of flow of substances between components. Further work on KRITIK is intended to expand that domain. Continued work on KRITIK will separate knowledge of components and substances from that of structure–behavior–function; such separation would resemble aspects of CADET's model.

Both CADET and KRITIK demonstrate computational models of devices that can be used in a case-based approach to compute design solutions for a limited class of engineering problems. For their limited domains, each can automatically produce design solutions and augment their respective case memories with these new solutions. However, they do not exhibit all of the characteristics that may be desired of a case-based system, e.g. storing failed design plans so that such plans can be avoided in the future [9]. Yet both yield insights into the complex issues that must be addressed when attempting to produce a system that can solve engineering design problems autonomously. On the other hand, the goal may be to produce a case-based system that assists a human designer with the design task by recalling previous solutions and alternatives. Indeed, one intended use of CADET is to 'brainstorm' design solutions for consideration by a designer. Other case-based systems supporting design allow varying degrees of human interaction; the next section focuses on several prototypes.

3.3. Case-based Design Assistants

Archie [36, 37] was an early prototype case-based system intended to aid the conceptual design of office buildings. Several issues were to be considered in the development of Archie, e.g. how to represent and organize design cases such that they are usable by a variety of participants in the conceptual design process, and how to integrate design cases with qualitative models capturing the dependencies between features of a building. The goals for Archie were to develop a system that could be used by real architects, contain a large memory of design cases and support a variety of design tasks; and, in so doing, to develop a theory of conceptual design aiding based on Archie, and to extract from Archie the means to develop a generic tool that could aid conceptual design in other problem domains as well.

The conceptual design task takes as input a specification of the goals and constraints for the office building to be designed. The architect describes these goals and constraints by selecting feature values, e.g. client organization type, frequency of visitors, total area, etc. Archie searches for relevant cases based on similarity to specified concept values and their importance. Cases with similarity values above a predetermined threshold are returned for consideration by the architect. The architect can then browse through the relevant cases to see how other architects solved similar problems. Solutions from several cases can be copied and combined to create a satisfactory design.

Archie was populated with approximately 20 cases; case information came from the implementers' experience with office buildings, from architectural journals and from post-occupancy evaluations written by architects identifying buildings' problems and suggestions for corrections. Archie's case memory included representations for primitive architectural concepts, domain models and the design cases themselves. Primitive concepts represent the objects, relations and parameters describing office buildings and provide the vocabulary for representing and indexing design cases. Concepts are organized hierarchically to specify the goals, plans and outcomes of design cases. Domain models represent causal relations between case concepts and provide the domain knowledge about office building design. During the design process, the architect can critique aspects of a partially specified conceptual design by investigating the domain models associated with that aspect. These models reveal how features of a design case interact and are linked to descriptions of design goals, plan comments, problem solutions and lessons learned.

Experience with Archie provided a number of insights into the issues associated with the development of case-based systems. Obtaining well-documented cases for office buildings was difficult; descriptions of original design goals and constraints were often unclear, case analyses were incomplete, and justifications for design decisions and outcomes were typically unavailable. Ideally, a well-documented case would contain a great deal of information, but the effort to gather such quantities of information is significant. From a user interface perspective, the case information presented to the user must be relevant to the user's interest, e.g. information pertinent when a conceptual design is being synthesized may not be relevant when a design is being critiqued. Providing case information at appropriate levels of detail was also problematic – it is necessary to organize case information at various levels of abstraction for easier comprehension by the user.

Archie-II [38] is a follow-on effort to the Archie project and strives for a better match between the demands of the design problem and the technology available. The aim is to build a relatively simple case browser that uses common graphic forms to organize and present interesting pieces of building designs when they are relevant to a designer's interests. Three aspects of building designs are identified to organize the useful pieces of building designs: design issues, physical locality or structural pieces, and functional systems. Design issues include such factors as cost or relationship to surroundings. Physical locality and structural pieces of buildings provide information about aspects such as siting, space organization and the like. Decomposition according to functional systems (e.g. electrical, plumbing, heating/ventilation/air conditioning) corresponds to the perspectives relevant to specialized construction issues. In essence, categorization according to the three aspects supports consideration of design solutions, potential problems or opportunities, and evaluation criteria according to an organization already familiar to an architect.

Presentation of cases in Archie-II addresses issues of case content, case abstraction and case relevance. Case content is intended to be dependent on how the designer is expected to use the information. Each piece of a case describes the situation it addresses, the solution carried out and the results of the solution. Each piece is thought of as a lesson, i.e. lessons that teach how to accomplish something and those that inform the user of considerations to be aware of. Lessons are narrative in form and may have an appropriate graphic associated. Lessons indicate a design guideline, its justification and the principle illustrated. Graphics annotated with such case information organize the presentation of information to the user and provide the means for exploration of relevant information to various levels of abstraction, e.g. building overview information associated with a picture of the building's exterior, a floorplan with annotations which are revealed through graphics interaction. The possibility for supplementing these presentation mechanisms with video and audio is also considered, thereby creating a hypermedia system which provides navigation techniques appropriate to the presentation context.

Archie-II is still under development. While it addresses many of the issues that were identified during development of Archie, there is still the significant task of populating the case memory with design cases. For office building design, the task is clearly formidable; however, for a much narrower design domain, the task of acquiring case information may be less so. An example of a system operating in a much narrower design domain is The Linkage Assistant (TLA); in its domain, the case memory can actually be computed automatically [39].

TLA is a case-based design system developed for the solution of mechanism design problems which can be solved using four-bar linkages. The case memory for TLA consists of a catalog of over 10,000 linkage design examples. Each linkage in the catalog is characterized by six parameters; the catalog was generated by stepping the parameters in a logarithmic fashion to cover a universe of designs that are considered practically realizable. The curve traced by a specified point on the linkage (i.e. the coupler curve) is associated with each linkage and is both qualitatively and quantitatively described. The coupler curves themselves are hierarchically organized into 256 families of curves according to their shape.

Linkage designs can be retrieved from TLA in one of two ways. The first is by manual browsing of the coupler curve families. TLA provides a graphic interface which supports the curve browsing process. The second method of retrieval is through a programmatic interface. A designer can write a query using TLA's programmatic interface which finds coupler curves according to specified qualitative and/or quantitative characteristics. Once potentially useful coupler curves have been retrieved, TLA allows the user to numerically optimize the characteristics of the linkage to suit the problem at hand. The user specifies the optimization through graphic controls and results are presented graphically as well. Additional work on TLA will focus on improving the coupler curve retrieval process through automatic generation of catalog queries based on a problem specification.

TLA allows satisfactory linkage design solutions to be synthesized rapidly and efficiently. It enables even novice linkage designers to achieve results that are of the same quality as that which would be expected from a more experienced designer. Therein lies the appeal of a case-based design assistant; to make the design process more efficient and to provide designers with the benefit of experience from existing design solutions.

Another system demonstrating the utility of the case-based assistance for design is SUPPORT which is used in the domain of elevator design [40]. SUPPORT provides assistance to a designer over a range of conceptual design processes: transforming customer requirements for elevator characteristics and behavior into high-level specifications, developing functional descriptions from those specifications and selecting components providing those functions. Its case memory contains approximately 200 cases which are each organized according to specifications and functions at various levels of abstraction. The case memory is indexed according to conceptual relations between functions. The designer works interactively with SUPPORT during each stage of the conceptual design process; cases can be retrieved in the applicable context at each stage. Sub-cases exhibiting functional characteristics similar to a current concept can be adapted by the designer. Newly adapted designs are stored in the case memory and automatically organized into the existing functional abstraction hierarchy with the designer's approval. Initial results with SUPPORT show that it is very helpful to the designer and can yield significant savings in time and cost.

Providing asistance over an even broader range of processes was the goal for the Episodal Associative Memory in the Rapid Design System (RDS) [41]. Here the intention was not only to provide relevant design solutions for mechanical product design but also to provide manufacturing planning information. In the RDS, a designer would manually develop a feature-based representation of product shape using a feature-based CAD system. From the tentative design representation, the case memory would be searched to retrieve similar product designs. Similarity matching was based on the description and location of designated features in the tentative design. New designs could be added to the case memory and organized according to feature characteristics. With RDS the designer would have access to existing detailed designs which may be sufficiently similar in form to a partially specified design to achieve the current design goal. Since fabrication plans were also to be associated with the detailed designs in the case memory, a ready-made manufacturing process plan could potentially be used or adapted, and previously identified fabrication problems could be avoided. This aspect of the RDS is potentially powerful, particularly with respect to integration of design and manufacturing. Unfortunately, it is not clear how much of the overall system was implemented. Also the notion of retrieving designs based on geometric form feature similarity appears to serve only the final processes of detailed design, but conceptual design not at all.

Thus far, a sampling of systems has been discussed that supports aspects of the design process for artifacts such as structures and mechanisms. One final system will be considered – this assists not with design of an artifact but with design of a configuration of artifacts for autoclave processing.

Clavier [11, 42] is a shop-floor assistant for autoclave curing of parts made from composite materials. Owing to the nature of the autoclave itself (e.g. uneven heating), different part heating rates and the desire to maximize throughput while maintaining quality results, the process of determining what configuration of parts will result in a successful load is complex. Traditionally, autoclave operators were given a prioritized list of parts to be cured; the operators would then resort to drawings of previously successful autoclave part configurations to find a configuration that included most of the high-priority parts. The part configuration would then be adapted by the operators to include more parts from the current list to be cured. Clavier was developed to help the autoclave operators pick successful part configurations using case-based techniques.

With Clavier, an operator specifies the parts needing to be cured along with a priority for each part. Clavier then searches its case memory for layouts that minimize the number of parts not on the list, maximize the number of high-priority parts on the list and maximize the total number of parts in the load. The case retrieval process results in a list of fully or partially instantiated layouts in order of the number and priority parts each yields. The operator then selects from the layouts according to their scores and the production statistics maintained for each layout.

For situations where initial case retrieval acquires layouts with parts not on the current parts list, Clavier searches for compatible substitutions for those parts. Parts are considered for substitution based on both global and local criteria. Considerations such as whether a part is of the same material as the rest of the load constitute global criteria. Local criteria include issues such as location of the part in the autoclave and types of surrounding parts. The operator has the ultimate decision as to which (if any) suggested substitutions should be allowed; using a graphical layout editor, the operator can modify the layout as desired or create entirely new layouts. After autoclave processing, the operator identifies whether or not an adapted case was successful; both successful cases and failed cases are stored in the case memory. Clavier retrieves only successful cases; however it uses the failed cases to predict whether adapted cases will be successful.

Clavier's initial memory of 20 cases has expanded to approximately 150 since it became operational in 1990. It retrieves a fully instantiated case 90% of the time and is expected to achieve nearly expert-level retrieval as the case memory continues to grow. Unfortunately, operators do not typically make use of automatically adapted cases because they are deemed unreliable; instead the cases are manually adapted by the operators. Therefore, Clavier's learning is due to the operators' expertise rather than its own reasoning capabilities. Nevertheless, Clavier demonstrates the power of a case-based approach in its abilities to incrementally learn in an evolutionary environment, reuse expert knowledge and allow even a novice to perform at the level of a more experienced operator.

4. Bridging the Gap: Research to Practice

It should be evident that there is a wide gap between the capabilities pursued in experimental systems enabling variant design and those available in today's CAD systems. The ability to perform detailed design of an artifact on a CAD system is well established, but provision of capabilities facilitating the task of transforming design goals, constraints and performance specifications into a conceptual design is virtually ignored.

Variant design lies at the boundary between conceptual design and detailed design. A designer can seek inspiration for a solution from existing designs satisfying one or more of the current design criteria. A designer may have a conceptual solution in mind but seeks to leverage the experience embedded in existing detailed design solutions that are conceptually similar. A designer may have an overall idea of the structure and organization of a satisfying artifact but can finish the new design faster given a previously designed solution. The question is, how can today's CAD systems be augmented to provide variant design capabilities?

First and foremost, the amount of information represented about an artifact must be greatly expanded beyond representations for nominal shape and acceptable variations. Form features are not a panacea; at best they provide a common vocabulary for instantiating and recognizing shapes as well as a mechanism for ensuring that geometric forms maintain particular relationships with neighboring geometry. Form features do not convey information necessary to support variant design. Parametric relations and equations are surely a boon to those who perform detailed design on a CAD system by easing tasks of shape definition and by relating shape definition to engineering rationale; but such parametric relations result from engineering decisions and do not provide much insight into the decision process itself. [Attempting to determine engineering rationale solely from stated parametric relations is akin to trying to determine a patient's medical history solely from a list of all medications prescribed.]

Migration of current CAD systems to a variant design environment can be done incrementally. Looking back at the lessons learned from Archie and the approach adopted for Archie-II, much can be gained by adding textual annotations for the goals that a design fulfills, the specifications that constrain the design process, the alternatives considered during the design process, the decisions taken and the justification for those decisions. Connecting representations for such textual annotations to the design models used in contempoary CAD systems would enable a designer to maintain an electronic record of important aspects of the design process. This design history would assist later designers in redesign tasks by allowing them to understand the context for a design and to make more informed decisions about how to modify a design. While textual annotations may not be considered the most efficient means for capturing design history information, they at least provide a semblance of a variant design environment.

A step further would be to provide representations that organize design data and enable directed retrieval of designs based on specified indices. As shown in systems like CADET and KRITIK, these organizing representations need to address concepts at a much higher level than artifact shape. The ability to represent concepts at various levels of abstraction increases the likelihood of finding relevant designs, as well as making the retrieval of designs more efficient. The indices necessary to retrieve relevant designs are directly related to these abstract representations since they provide the vocabulary for retrieval. Determining what model should underlie such representations (e.g. function–behavior–structure) and how current CAD systems could be augmented with a model applicable to the domain of mechanical artifacts is a problem meriting further investigation. A joint effort composed of vendors, users and researchers, such as being considered by the NCMS RRM consortium, could have a significant impact on this problem area [43].

The user interface facilities necessary to support variant design with CAD systems must also be addressed. The need to capture large amounts of information corresponding to design history annotations, functional characterizations, and the like, must be balanced with the amount of additional effort imposed on designers and measured against the perceived benefits. Again considering Archie-II's approach, interaction with a system must be meaningful to the user's context and can be enhanced by a combination of text, graphics, audio, simulations, and so on. Thus an annotation describing the aesthetic constraints for materials selection may be most easily captured as audio. A simulation showing the assemblage of a standard fixture to a flange on a part may be a good way to illustrate the rationale for an otherwise unnecessary flange. The capabilities of today's computer systems for capturing and replaying such data should not be left unexploited by the vendors supporting the engineering community.

4.1. Standards Revisited

A Parts Library series of standards is under development in ISO. The Parts Library specifications are intended to facilitate exchange of files containing standard parts and also allow implementation of shared databases of parts library data [44]. The Parts Library specifications are being developed in conjunction with STEP; STEP would provide many of the fundamental representation and description specifications used by the Parts Library specifications. From the viewpoint of the Parts Library specification, standard parts may be considered as parts that are common, off-the-shelf components used in industry (e.g. fasteners), but may also be parts that are standardized internally by a particular company.

A particularly interesting aspect of the Parts Library standards is the accommodation for description of multiple functional models of a part. These functional models are intended to provide mechanisms for a CAD system to generate different information representations for particular functional aspects of each standard part so described. For example, consider a fastener that is primarily characterized by a few numeric attributes. A particular functional model associated with the fastener's characterization might allow a CAD system to activate a method provided by the functional model that displays a particular graphic view of the fastener. In this example, the CAD system would be relieved of the task of generating its own geometric model of the fastener from the fastener's simple characterization.

Recall the discussion above about abstract models for mechanical artifacts. With a useful abstract model for mechanical artifacts, the functional modeling capabilities envisioned for the standard parts library specifications could be used to convey abstract models of standard parts and to compute results from the abstract models (e.g. to compute functional indices as in KRITIK). While this may not seem particularly useful for simple parts such as fasteners, the standard parts library facility does not restrict the modeling domain to simple parts. For company-internal use, the standard parts library facility could be used as the vehicle for a company-specific case memory accessible by a variety of in-house systems.

5. Conclusion

Computational systems that autonomously solve design problems may exist in the future, but for the near-term, human designers solve design problems and devise their solutions using computer aids. Since their inception 30 years ago, the computer-based tools that are employed in the design process have progressed dramatically in terms of geometric coverage, analysis capabilities, accuracy, visualization capabilities and speed. The 'look and feel' of the design process has changed along with the availability of these tools, yet many intrinsic aspects of the design process itself remain the same. The fact that designers will need to redesign existing designs, or use existing designs as the basis for a new design, or gain insights from the knowledge captured in an existing design, has not changed – nor is it likely that it ever will.

This paper has examined the feature-based and parametric modeling capabilities of current CAD systems and the information standards relevant to those capabilities from the perspective of how such technologies support variant design. It can be concluded that those capabilities support variant design only in the sense that they may ease aspects of redesign but do not by themselves make for an environment enabling variant design. The examination of recent research approaches in analogical reasoning and case-based design illustrates the complex issues that need to be considered for autonomously retrieving and applying existing designs to solve new design problems. Those research efforts provide a vision of what capabilities may be achieved in the future and have yielded insight into what can be done in the short-term. From those efforts, the conclusion can be drawn that current CAD systems could soon be augmented with techniques identified in case-based design research and that doing so would provide great benefit to designers.

Acknowledgments

The National Center for Manufacturing Science's (NCMS) Rapid Response Manufacturing (RRM) consortium is funded in part by an award from the NIST Advanced Technology Program (ATP). Participants in the RRM consortium include Ford Motor Company, General Motors, Martin Marietta Energy Systems, Texas Instruments and United Technologies. This report was prepared under the auspices of NIST's ATP intramural funding.

References

1. ISO/DIS 10303–1 (1993) Industrial automation systems and integration – Product data representation and exchange – Part 1: Overview and fundamental principles, International Organization for Standardization, Geneva, Switzerland
2. ISO/DIS 10303–42 (1993) Industrial automation systems and integration – Product data representation and exchange – Part 42: Integrated generic resources: Geometric and topological representation, International Organization for Standardization, Geneva, Switzerland
3. ISO/WD 10303–48 (1992) Industrial automation systems and integration – Product data representation and exchange – Part 48: Intregrated generic resources: Form features. TC184/SC4/WG3 Document N102, International Organization for Standardization, Geneva, Switzerland
4. Technicom, Inc. (1992) Parametric, Variational, and Feature Based Modeling Study Report, Clifton, NJ
5. Kling, R.E. (1971) A paradigm for reasoning by analogy, Artificial Intelligence, 2, 147–178
6. Simpson, R.L. (1985) A computer model of case-based reasoning in problem solving: an investigation in the domain of dispute mediation, PhD Thesis, Tech. Rept. GIT-ICS-85/18, Georgia Institute of Technology, Atlanta, GA
7. Kriegsman, M.; Barletta, R. (1993) Building a case-based help desk application, IEEE Expert, 8, 6, 18–26
8. Simoudis, E. (1992) Using case-based retrieval for customer technical support, IEEE Expert, 7, 5, 7–12
9. Hammond, K.J. (1989) Case-Based Planning: Viewing Planning as a Memory Task, Academic Press, San Diego, CA
10. Hinrichs, T.R.; Kolodner, J.L. (1991) The roles of adaptation in case-based design, Proceedings of the Ninth National Conference on Artificial Intelligence, Anaheim, CA, 28–33
11. Hennessy, D.; Hinkle, D. (1992) Applying case-based reasoning to autoclave loading, IEEE Expert, 7, 2, 21–26
12. Carbonell, J.G. (1986) Derivational analogy: A theory of reconstructive problem solving and expertise acquisition, Machine Learning: An Artificial Intelligence Approach, Vol. II, R.S. Michalski, J.G. Carbonell, T.M. Mitchell (Editors), Morgan Kaufman, Los Altos, CA, 371–391.
13. Hall, R.P. (1989) Computational approaches to analogical reasoning: a comparative analysis, Artificial Intelligence, 39, 39–120
14. Evans, T.G. (1968) A program for the solution of a class of geometric analogy intelligence test questions, Semantic Information Processing, M. Minsky (Editor), MIT Press, Cambridge, MA, 271–353
15. Carbonell, J.G. (1983) Learning by analogy: formulating and generalizing plans from past experience, Machine Learning: An Artificial Intelligence Approach, R.S. Michalski, J.G. Carbonell, T.M. Mitchell (Editors), Tioga Press, Palo Alto, CA, 137–161
16. Munyer, J.C. (1981) Analogy as a means of discovery in problem solving and learning, PhD Thesis, University of California at Santa Cruz, CA
17. Carbonell, J.G. (1981) Invariance hierarchies in metaphor interpretation, Proceedings of the Third Annual Meeting of the Cognitive Science Society, Berkeley, CA, 292–295
18. Kedar-Cabelli, S.T. (1985) Purpose-directed analogy, Proceedings Seventh Annual Conference of the Cognitive Science Society, Irvine, CA, 150–159
19. McDermott, J. (1979) Learning to use analogies, Proceedings IJCAI–79, Tokyo, 568–576
20. Winston, P.H. (1980) Learning and reasoning by analogy, Communication of the ACM, 23, 12, 689–703
21. Navinchandra, D.; *et al.* (1987) Analogy-based engineering problem solving: an overview, Artificial Intelligence in Engineering: Tools and Techniques, D. Sriram, R.A. Adey (Editors), Computational Mechanics Publishers, Southampton, UK, 273–285
22. Maher, M.I.; Zhao, F. (1987) Using experience to plan the synthesis of new designs, Expert Systems in Computer-Aided Design, J. Gero (Editor), North-Holland, Amsterdam, 349–367
23. Zhao, F.; Maher, M.L. (1988) Using analogical reasoning to design buildings, Engineering with Computers, 4, 107–119
24. Daube, F.; Hayes-Roth, B. (1989) A case-based mechanical redesign system, Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, Detroit, MI, 1402–1407
25. Mostow, J. (1989) Design by derivational analogy: issues in the automated replay of design problems, Artificial Intelligence, 40, 119–184
26. Wile, D.S. (1983) Program developments: formal explanations of implementations, Communications of the ACM, 26, 11, 902–911
27. Steinberg, L.I.; Mitchell, T.M. (1985) The redesign system: a knowledge-based approach to VLSI CAD, IEEE Design & Test, 2, 45–54
28. Mostow, J.; Barley, M. (1987) Automated reuse of design plans, Proceedings 1987 International Conference on Engineering Design (ICED87), Boston, MA, 632–647.
29. Huhns, M.N.: Acosta, R.D. (1988) ARGO: a system for design by analogy, IEEE Expert, 3, 3, 53–68
30. Sycara, K.; *et al.* (1992) CADET: a case-based synthesis tool for engineering design, International Journal of Expert Systems, 4, 2, 157–188
31. Goel, A. (1992) Representation of design functions in experience-based design, Intelligent Computer Aided Design, D.C. Brown, M. Waldron, H. Yoshikawa (Editors), North-Holland, Amsterdam, 283–303
32. Goel, A.; Chandrasekaran, B. (1992) Case-based design: a task analysis, Artificial Intelligence in Engineering Design, Vol. II, C. Tong, D. Sriram (Editors), Academic Press, New York, 165–183
33. Sycara, K.; Navin chandra, D. (1989) Integrating case-based

reasoning and qualitative reasoning in engineering design, Artificial Intelligence in Engineering, J. Gero (Editor), Computational Mechanics Publications, Southampton, UK, 231–250

34. Sycara, K.; Navin chandra, D. (1992) Retrieval strategies in a case-based design system, Artificial Intelligence in Engineering Design, Vol. II, C. Tong, D. Sriram (Editors), Academic Press, New York, 145–163
35. Goel, A.; Chandrasekaran, B. (1989) Use of device models in adaptation of design cases, Proceedings of the Second DARPA Case-Based Reasoning Workshop, Pensacola, FL, 100–109
36. Goel, A.; *et al.* (1991) Towards a case-based tool for aiding conceptual design problem solving, Proceedings of the Third DARPA Workshop on Case-Based Reasoning, Washington, DC, 109–120
37. Pearce, M.; *et al.* (1992) Case-based design support: a case study in architectural design, IEEE Expert, 7, 5, 14–20
38. Domoshek, E.A.; Kolodner, J.L. (1992) Toward a case-based aid for conceptual design, International Journal of Expert Systems, 4, 2, 201–220
39. Kramer, G.A.; Barrow, H.G. (1992) A case-based approach to the design of mechanical linkages, Artificial Intelligence in Engineering Design, Vol. II, C. Tong, D. Sriram (Editors), Academic Press, New York, 443–466
40. Nakatani, Y.; *et al.* (1992) Engineering design support framework by case-based reasoning, ISA Transactions, 31, 2, 165–180
41. Pao, Y.; *et al.* (1992) The role of the episodal associative memory in feature-based design, Intelligent Computer Aided Design, D.C. Brown, M. Waldron, H. Yoshikawa (Editors), North-Holland, Amsterdam, 309–325
42. Barletta, R.; Hennessy, D. (1989) Case adaptation in autoclave layout design, Proceedings of the Second DARPA Case-based Reasoning Workshop, Pensacola, FL, 203–207
43. Warfield, J. (Editor) (1993) An Interactive Management Workshop on the Variant Design Process, Ford Motor Company, Dearborn, MI
44. ISO/CDC 13584–1 (1993) Industrial automation systems and integration – Parts Library – Part 1: Overview and fundamental principles. TC184/SC4 Document N205, International Organization for Standardization, Geneva, Switzerland